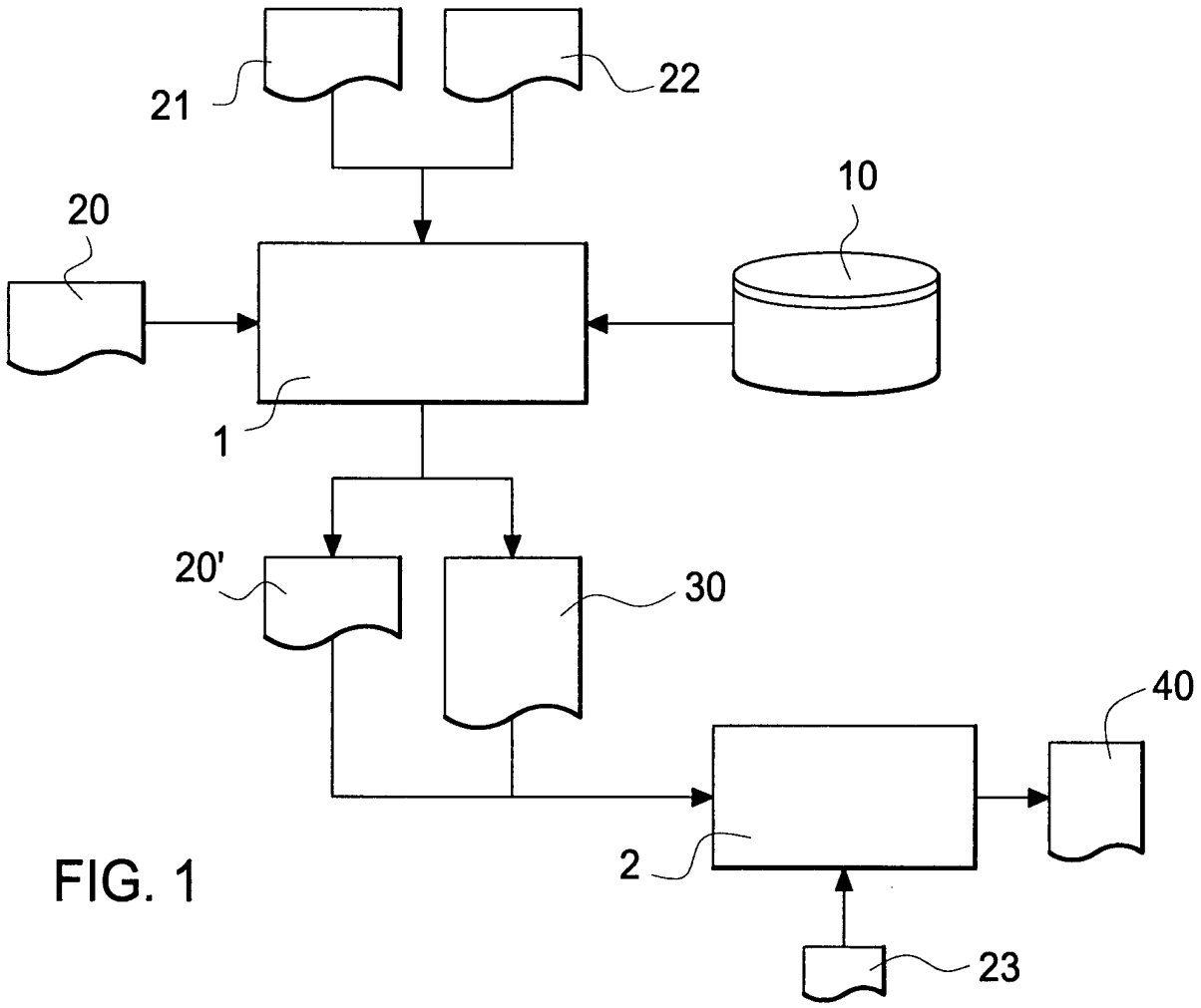


FIG. 1

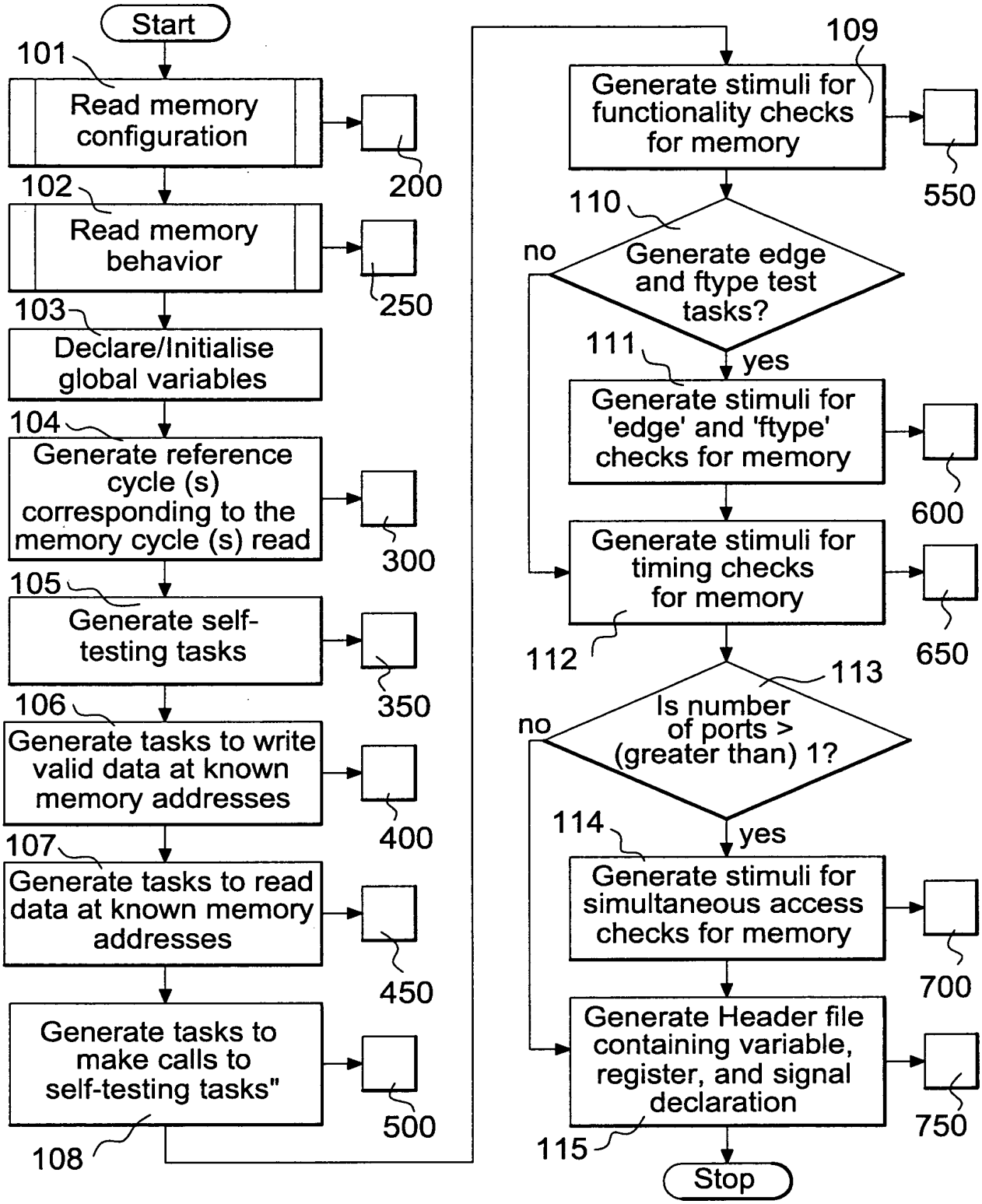

Start
101
Read memory configuration
200
102
Read memory behavior
250
103
Declare/Initialise global variables
104
Generate reference cycle (s) corresponding to the memory cycle (s) read
300
105
Generate self-testing tasks
350
106
Generate tasks to write valid data at known memory addresses
400
107
Generate tasks to read data at known memory addresses
450
Generate tasks to make calls to self-testing tasks"
500
108
109
Generate stimuli for functionality checks for memory
550
110
Generate edge and ftype test tasks?
no
yes
111
Generate stimuli for 'edge' and 'ftype' checks for memory
600
Generate stimuli for timing checks for memory
650
112
113
Is number of ports > (greater than) 1?
no
yes
114
Generate stimuli for simultaneous access checks for memory
700
Generate Header file containing variable, register, and signal declaration
750
115
Stop

FIG. 2

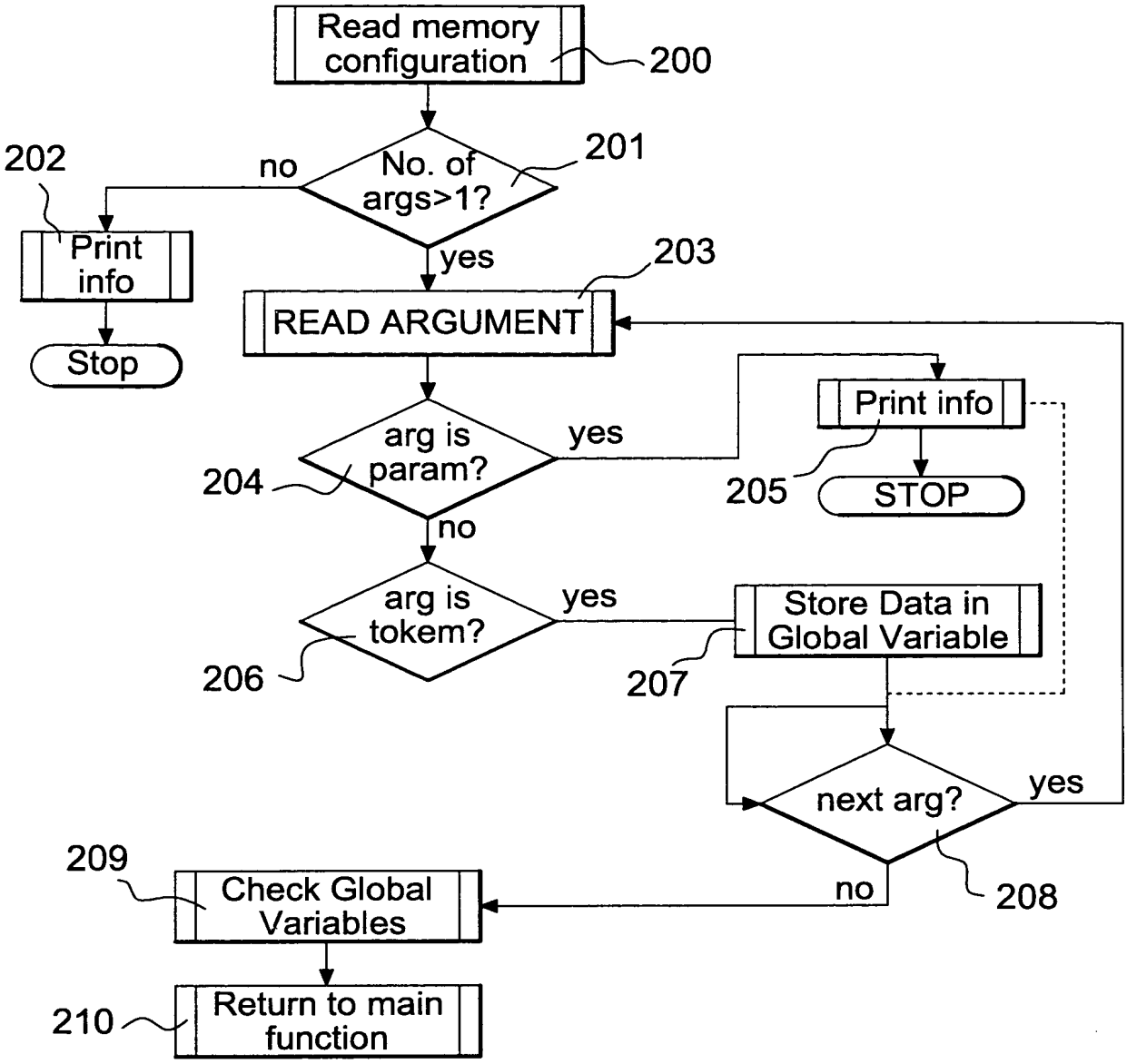


FIG. 3

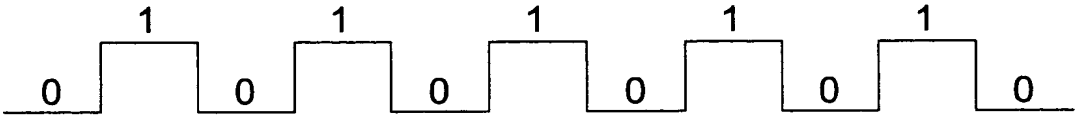


FIG. 6A

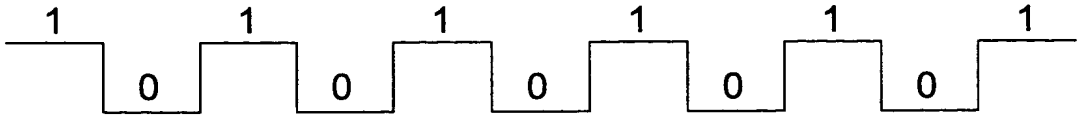


FIG. 6B

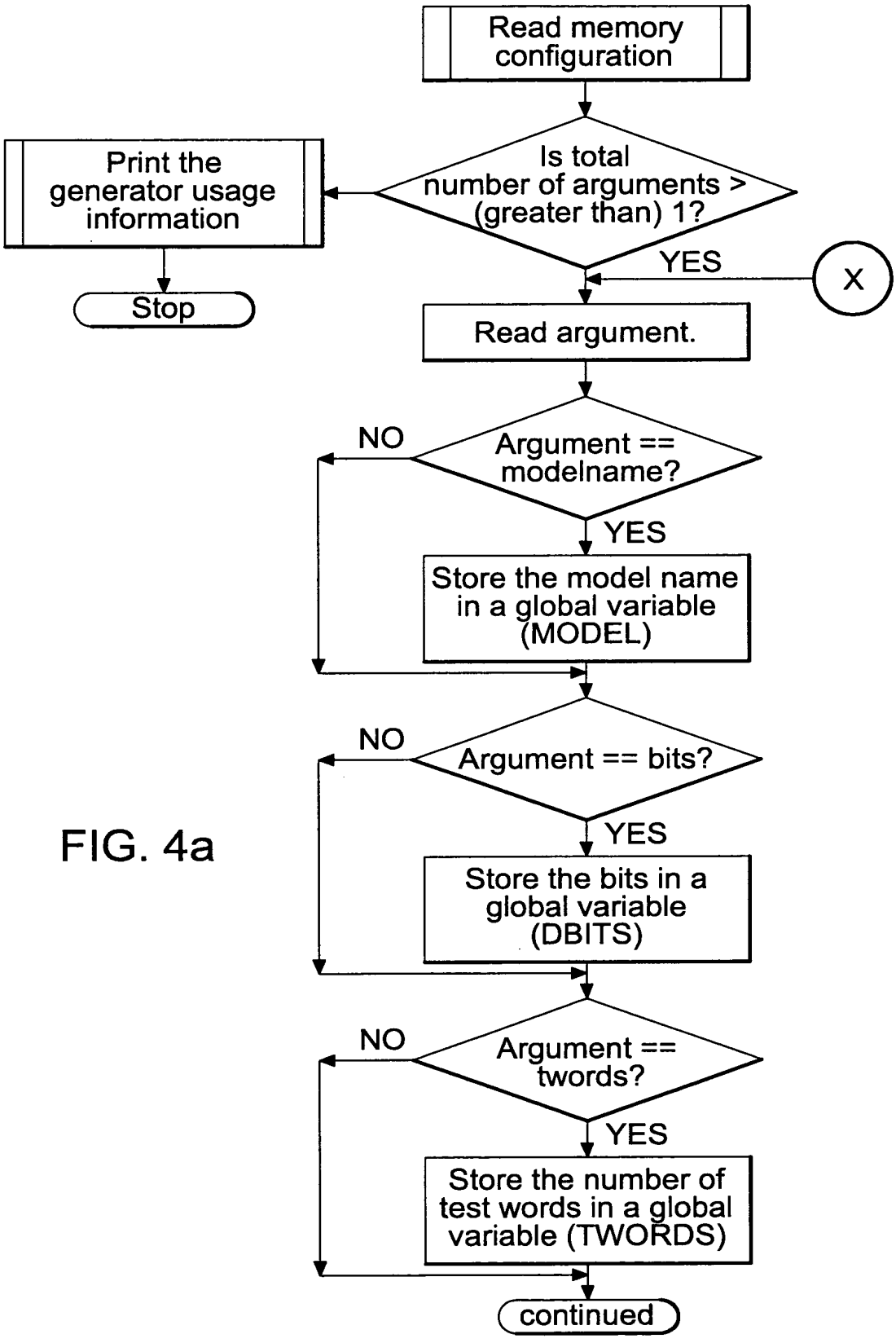

Read memory configuration
Is total number of arguments > (greater than) 1?
Print the generator usage information
Stop
YES
X
Read argument.
NO
Argument == modelname?
YES
Store the model name in a global variable (MODEL)
NO
Argument == bits?
YES
Store the bits in a global variable (DBITS)
NO
Argument == twords?
YES
Store the number of test words in a global variable (TWORDS)
continued

FIG. 4a

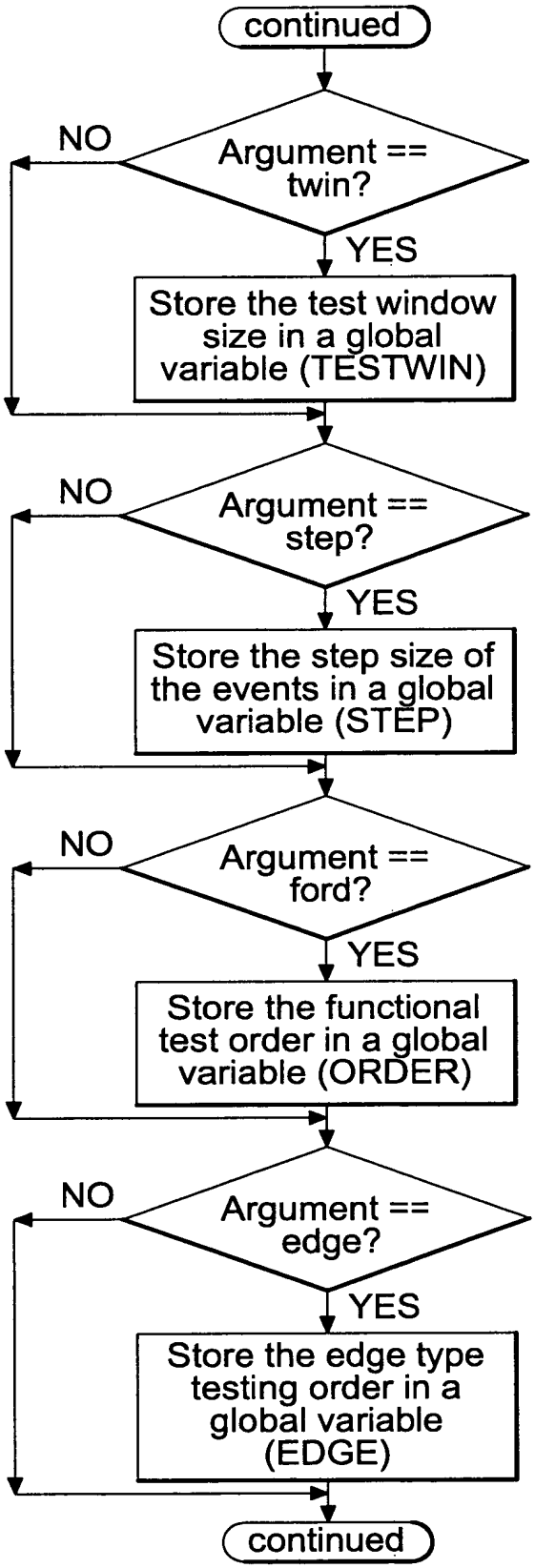

continued
Argument == twin?
NO
YES
Store the test window size in a global variable (TESTWIN)
Argument == step?
NO
YES
Store the step size of the events in a global variable (STEP)
Argument == ford?
NO
YES
Store the functional test order in a global variable (ORDER)
Argument == edge?
NO
YES
Store the edge type testing order in a global variable (EDGE)
continued

FIG. 4b

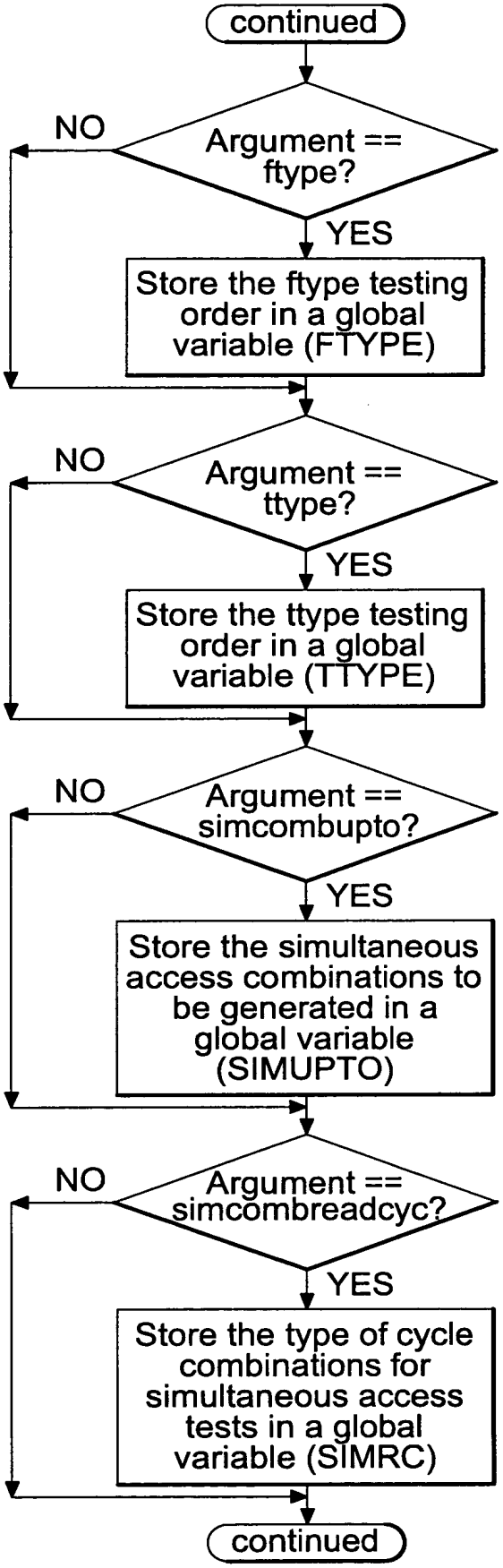

continued
Argument == ftype?
NO
YES
Store the ftype testing order in a global variable (FTYPE)
Argument == ttype?
NO
YES
Store the ttype testing order in a global variable (TTYPE)
Argument == simcombupto?
NO
YES
Store the simultaneous access combinations to be generated in a global variable (SIMUPTO)
Argument == simcombreadcyc?
NO
YES
Store the type of cycle combinations for simultaneous access tests in a global variable (SIMRC)
continued

FIG. 4c

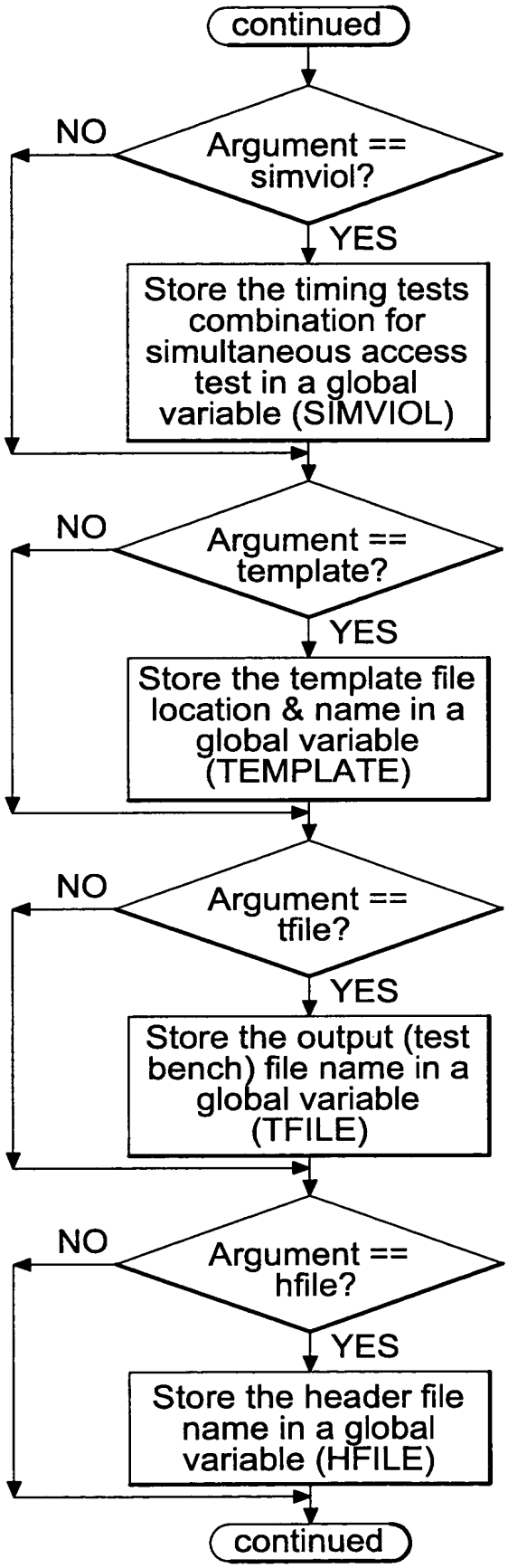

continued
Argument == simviol?
NO
YES
Store the timing tests combination for simultaneous access test in a global variable (SIMVIOL)
Argument == template?
NO
YES
Store the template file location & name in a global variable (TEMPLATE)
Argument == tfile?
NO
YES
Store the output (test bench) file name in a global variable (TFILE)
Argument == hfile?
NO
YES
Store the header file name in a global variable (HFILE)
continued

FIG. 4d

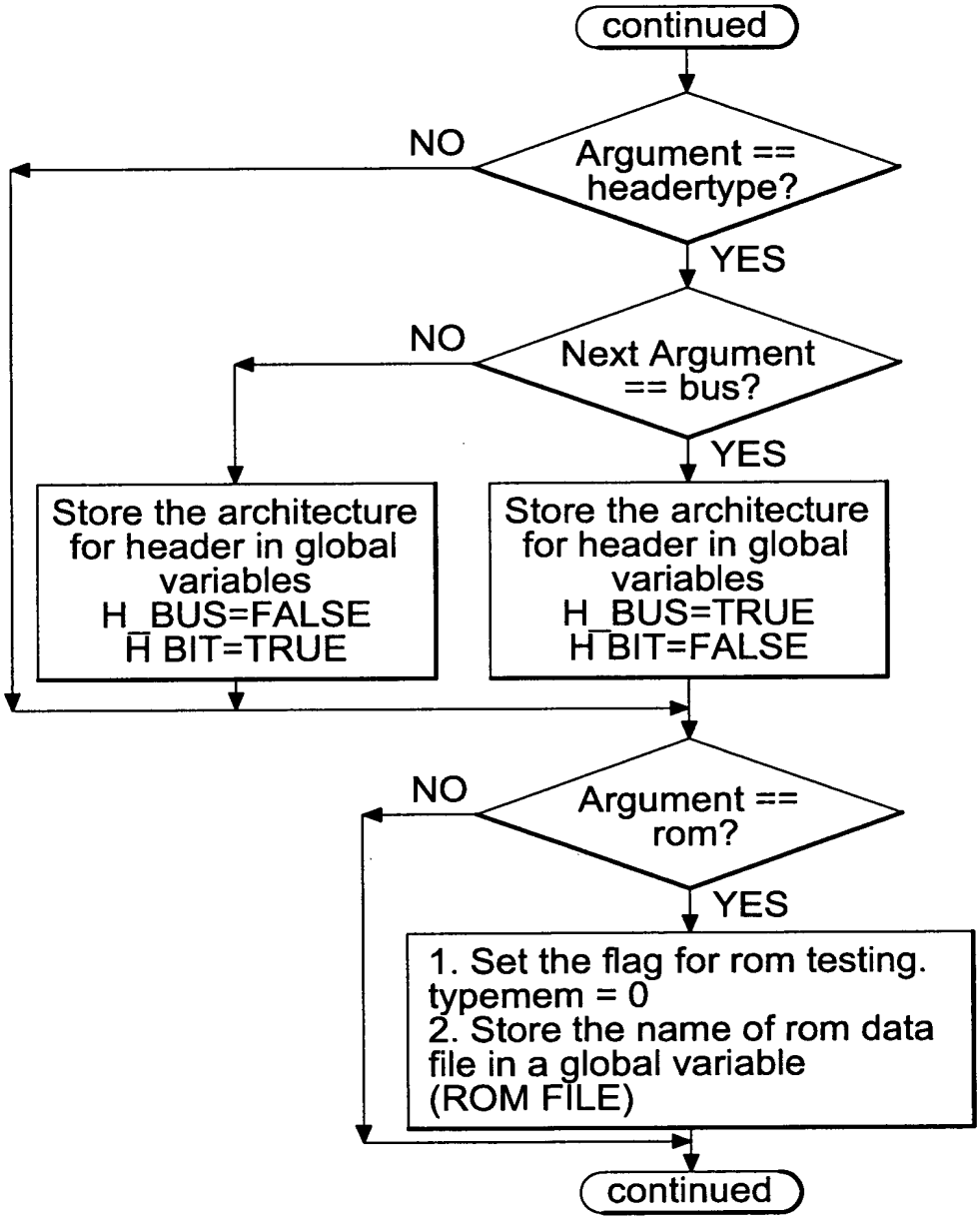
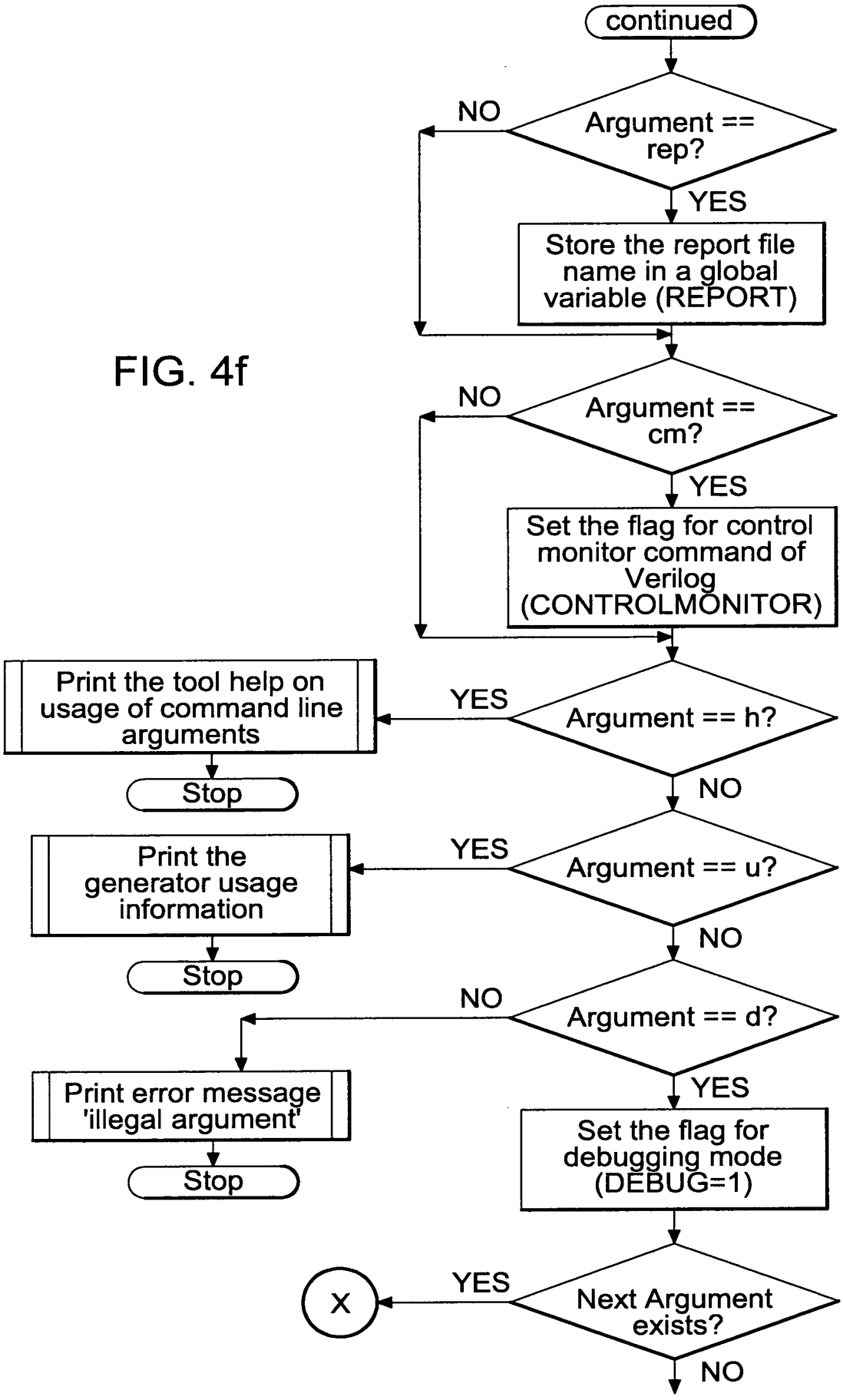

continued
Argument == headertype?
NO
YES
Next Argument == bus?
NO
YES
Store the architecture for header in global variables H_BUS=FALSE H_BIT=TRUE
Store the architecture for header in global variables H_BUS=TRUE H_BIT=FALSE
Argument == rom?
NO
YES
1. Set the flag for rom testing. typemem = 0
2. Store the name of rom data file in a global variable (ROM FILE)
continued

FIG. 4e

FIG. 4f


continued
Argument == rep?
NO
YES
Store the report file name in a global variable (REPORT)
Argument == cm?
NO
YES
Set the flag for control monitor command of Verilog (CONTROLMONITOR)
Argument == h?
YES
Print the tool help on usage of command line arguments
Stop
NO
Argument == u?
YES
Print the generator usage information
Stop
NO
Argument == d?
NO
Print error message 'illegal argument'
Stop
YES
Set the flag for debugging mode (DEBUG=1)
Next Argument exists?
YES
X
NO

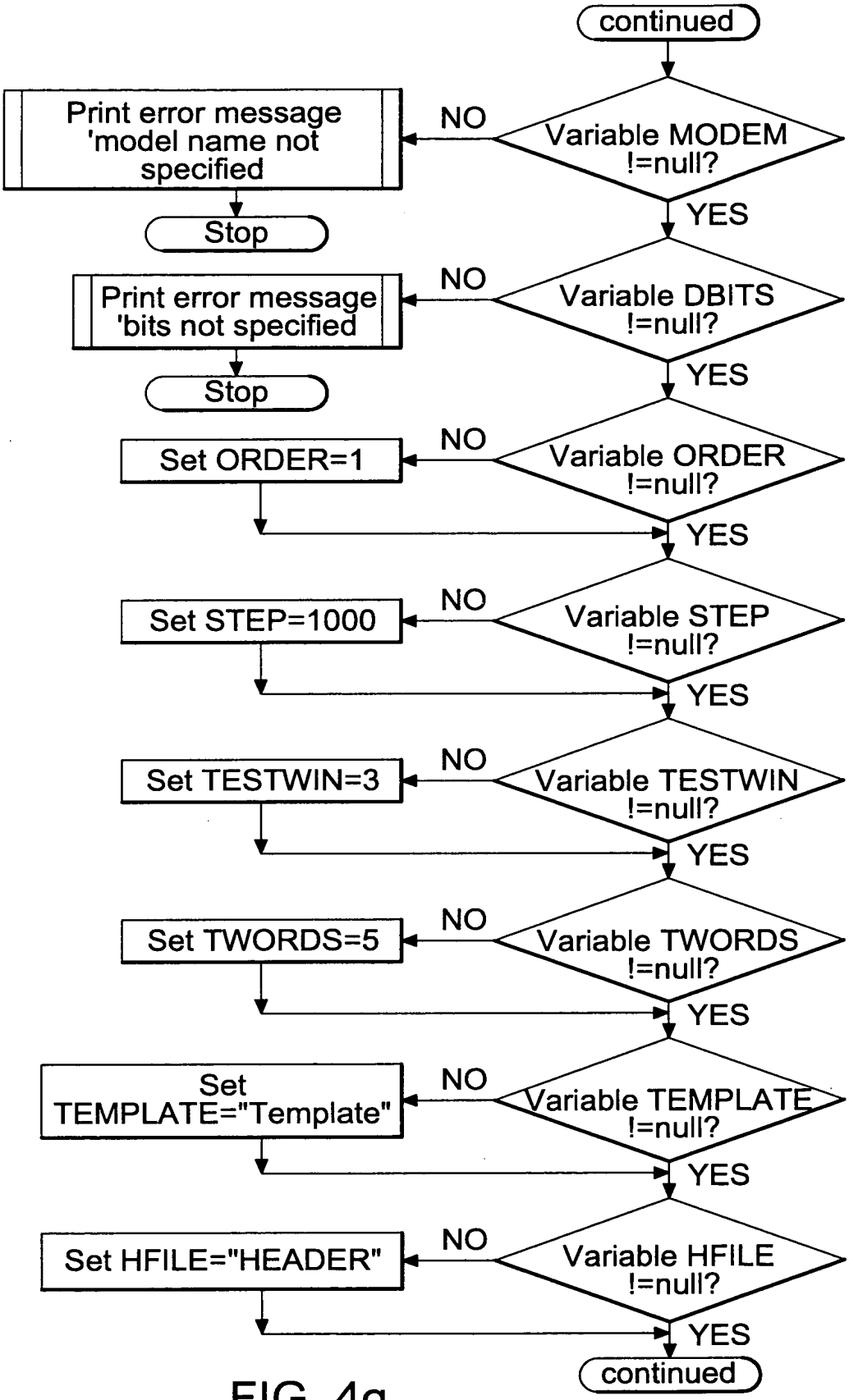

continued
Variable MODEM !=null?
NO
Print error message 'model name not specified
Stop
YES
Variable DBITS !=null?
NO
Print error message 'bits not specified
Stop
YES
Variable ORDER !=null?
NO
Set ORDER=1
YES
Variable STEP !=null?
NO
Set STEP=1000
YES
Variable TESTWIN !=null?
NO
Set TESTWIN=3
YES
Variable TWORDS !=null?
NO
Set TWORDS=5
YES
Variable TEMPLATE !=null?
NO
Set TEMPLATE="Template"
YES
Variable HFILE !=null?
NO
Set HFILE="HEADER"
YES
continued

FIG. 4g

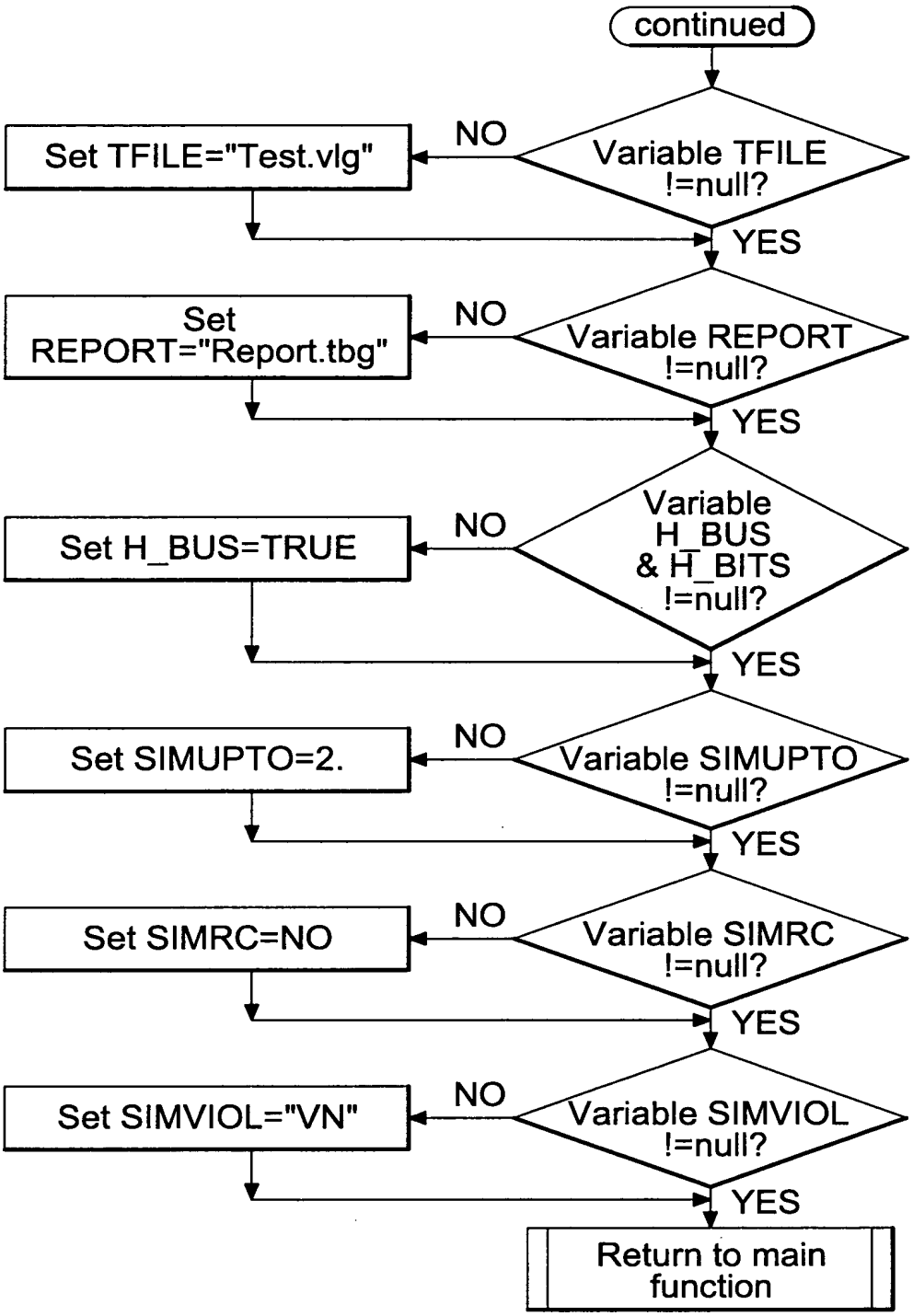

continued
Variable TFILE !=null?
NO
Set TFILE="Test.vlg"
YES
Variable REPORT !=null?
NO
Set REPORT="Report.tbg"
YES
Variable H_BUS & H_BITS !=null?
NO
Set H_BUS=TRUE
YES
Variable SIMUPTO !=null?
NO
Set SIMUPTO=2.
YES
Variable SIMRC !=null?
NO
Set SIMRC=NO
YES
Variable SIMVIOL !=null?
NO
Set SIMVIOL="VN"
YES
Return to main function

FIG. 4h

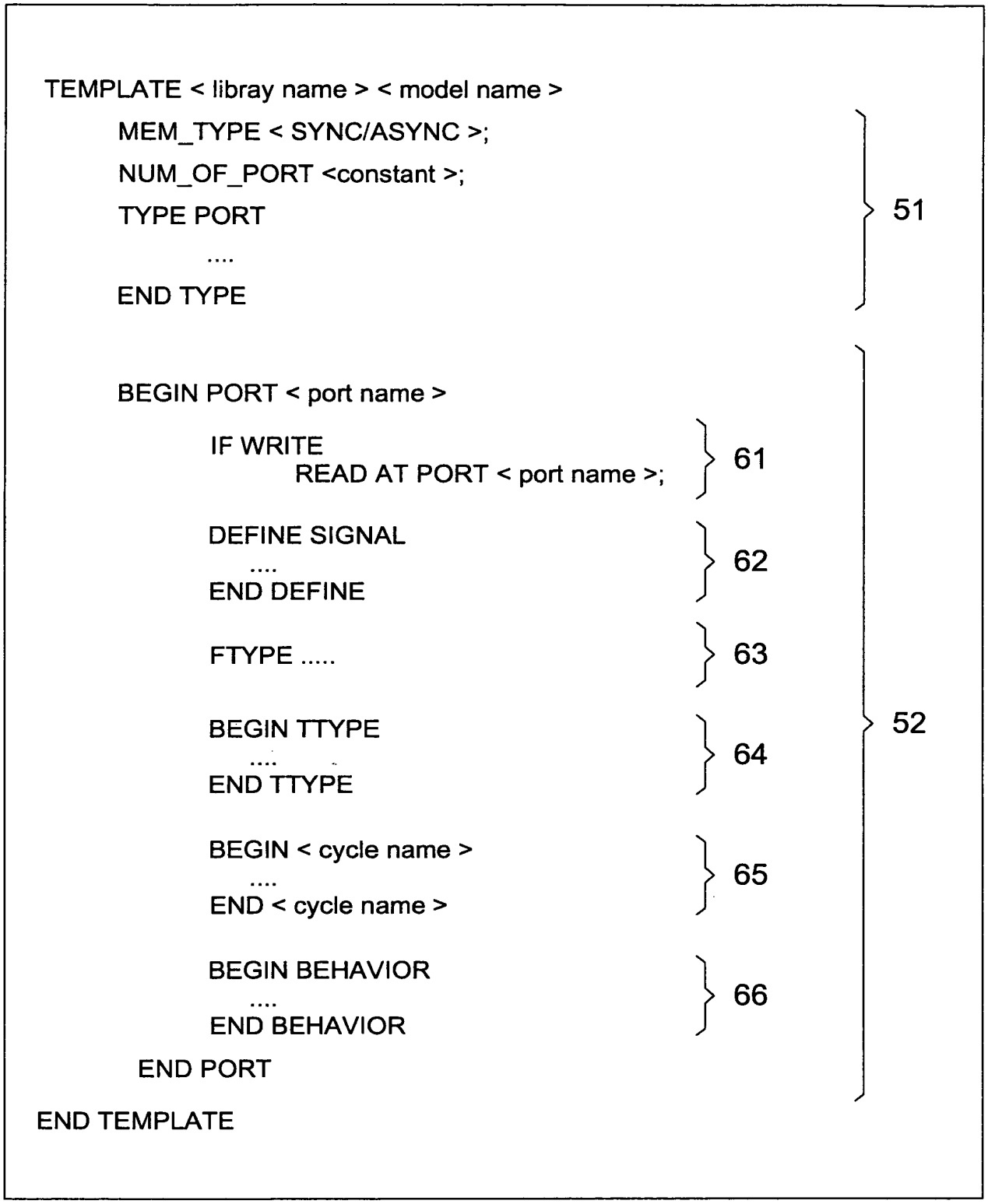

TEMPLATE < libray name > < model name >
MEM_TYPE < SYNC/ASYNC >;
NUM_OF_PORT <constant >;
TYPE PORT
....
END TYPE
51
BEGIN PORT < port name >
IF WRITE
READ AT PORT < port name >;
61
DEFINE SIGNAL
....
END DEFINE
62
FTYPE .....
63
BEGIN TTYPE
....
END TTYPE
64
BEGIN < cycle name >
....
END < cycle name >
65
BEGIN BEHAVIOR
....
END BEHAVIOR
66
END PORT
52
END TEMPLATE

FIG. 5

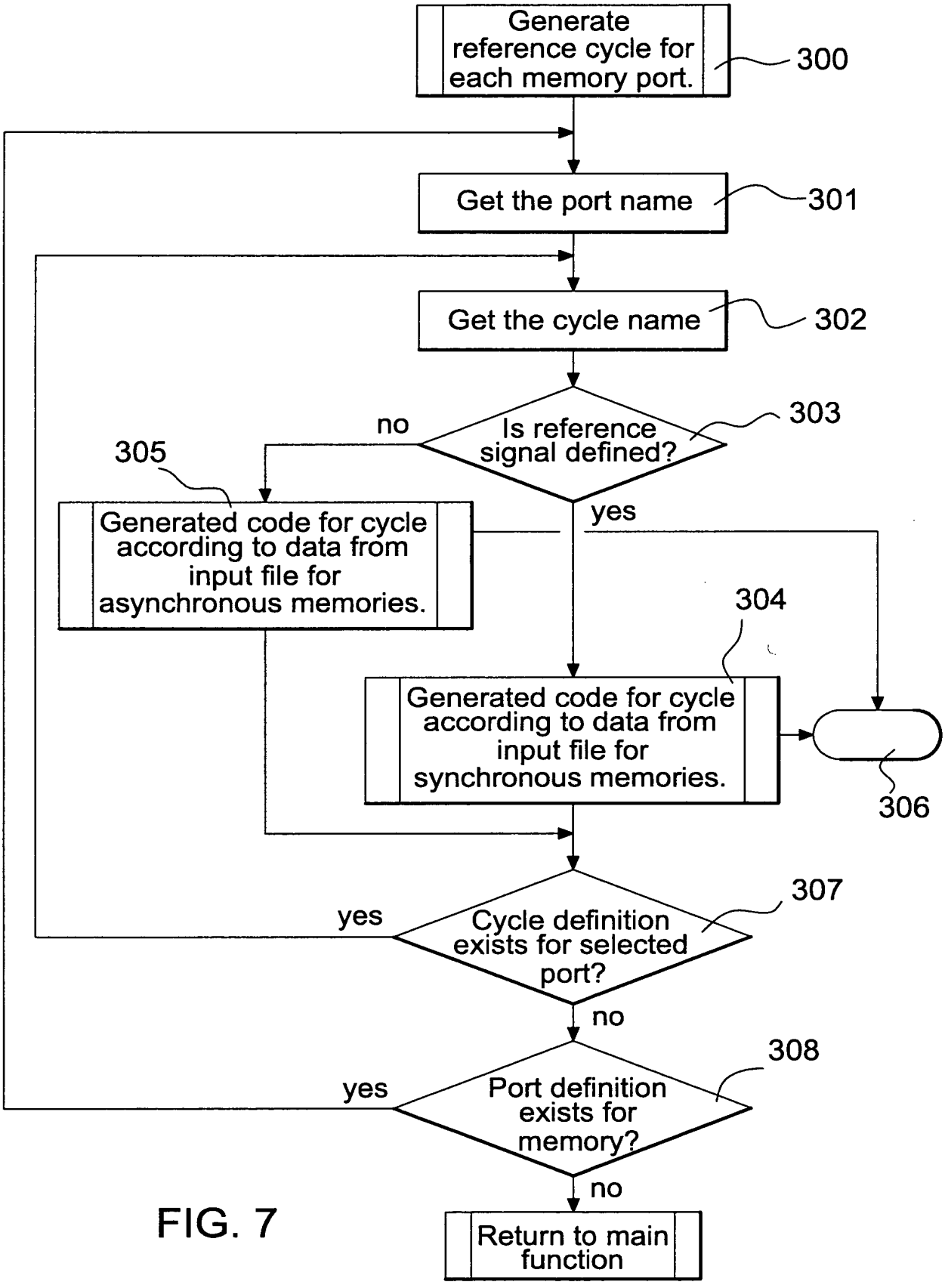

Generate reference cycle for each memory port.
300
Get the port name
301
Get the cycle name
302
Is reference signal defined?
303
no
yes
305
Generated code for cycle according to data from input file for asynchronous memories.
304
Generated code for cycle according to data from input file for synchronous memories.
306
Cycle definition exists for selected port?
307
yes
no
Port definition exists for memory?
308
yes
no
Return to main function

FIG. 7

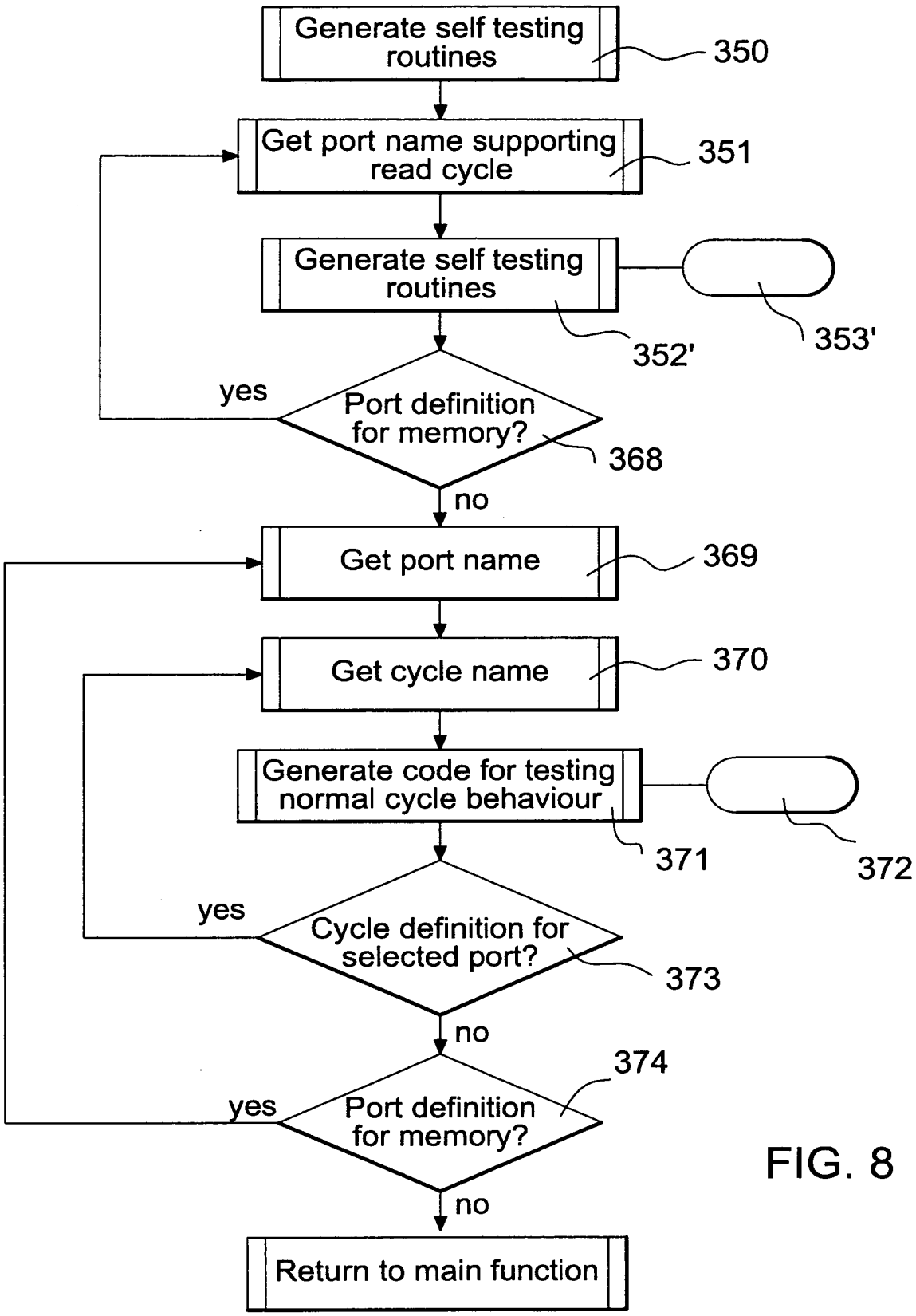

Generate self testing routines
350
Get port name supporting read cycle
351
Generate self testing routines
352'
353'
yes
Port definition for memory?
368
no
Get port name
369
Get cycle name
370
Generate code for testing normal cycle behaviour
371
372
yes
Cycle definition for selected port?
373
no
374
yes
Port definition for memory?
no
Return to main function

FIG. 8

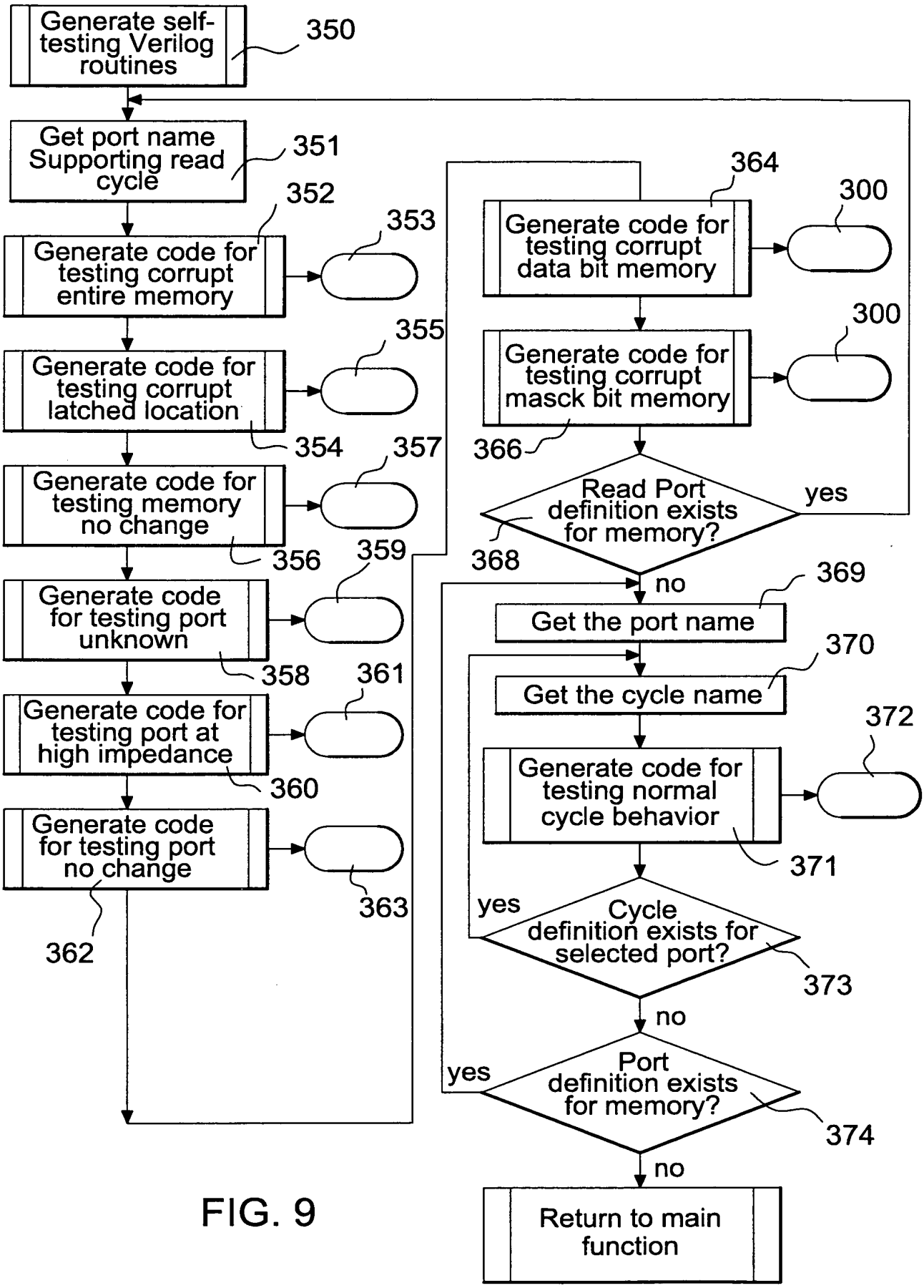

Generate self-testing Verilog routines
350
Get port name Supporting read cycle
351
Generate code for testing corrupt entire memory
352
353
Generate code for testing corrupt latched location
355
354
Generate code for testing memory no change
357
356
Generate code for testing port unknown
359
358
Generate code for testing port at high impedance
361
360
Generate code for testing port no change
363
362
Generate code for testing corrupt data bit memory
364
300
Generate code for testing corrupt masck bit memory
300
366
Read Port definition exists for memory?
yes
368
no
Get the port name
369
Get the cycle name
370
Generate code for testing normal cycle behavior
372
371
yes
Cycle definition exists for selected port?
373
no
yes
Port definition exists for memory?
374
no
Return to main function

FIG. 9

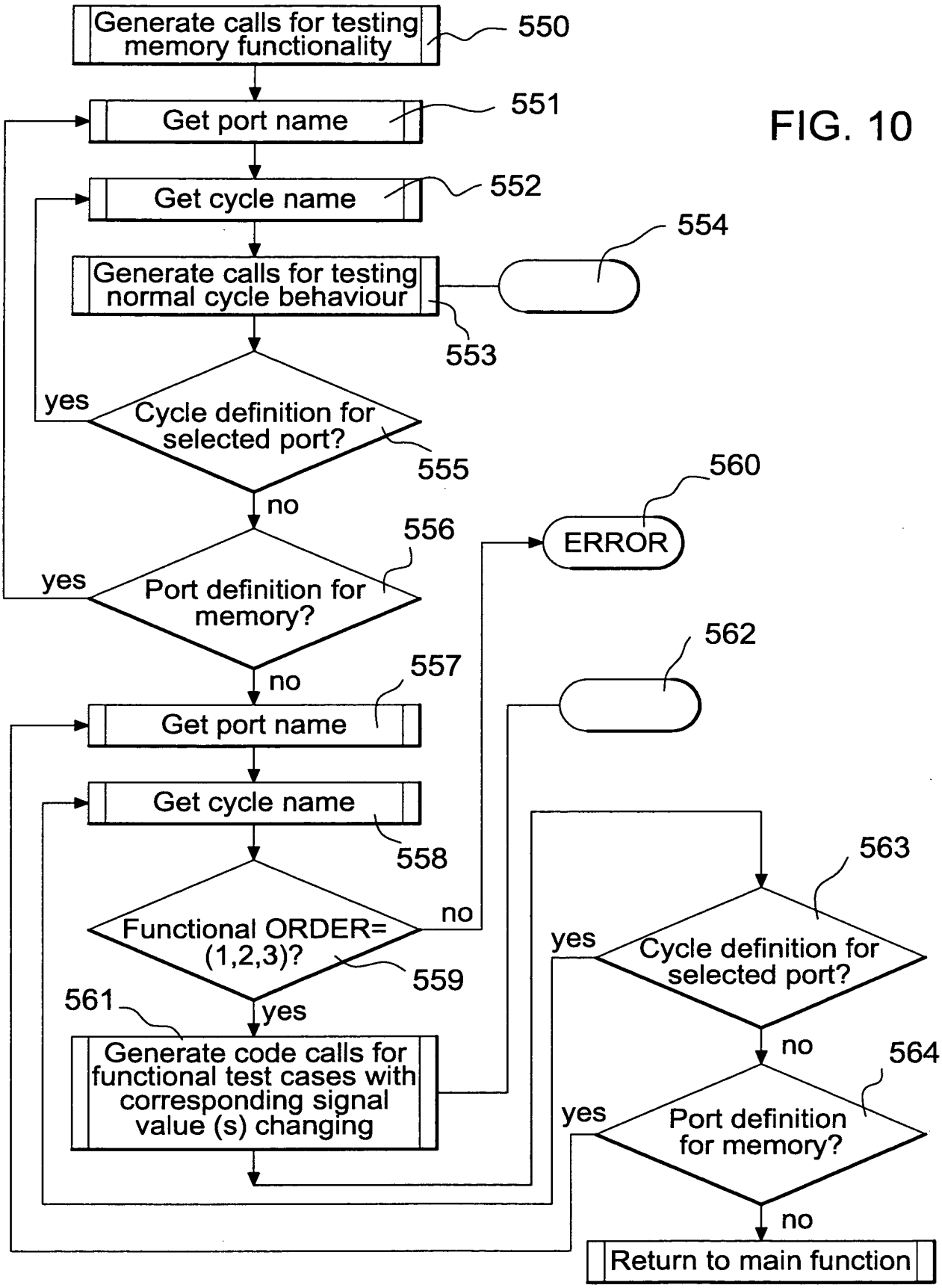

Generate calls for testing memory functionality
550
Get port name
551
Get cycle name
552
Generate calls for testing normal cycle behaviour
554
553
yes
Cycle definition for selected port?
555
no
560
ERROR
556
yes
Port definition for memory?
562
no
557
Get port name
Get cycle name
558
Functional ORDER= (1,2,3)?
no
559
563
yes
Cycle definition for selected port?
561
yes
Generate code calls for functional test cases with corresponding signal value (s) changing
no
564
yes
Port definition for memory?
no
Return to main function

FIG. 10

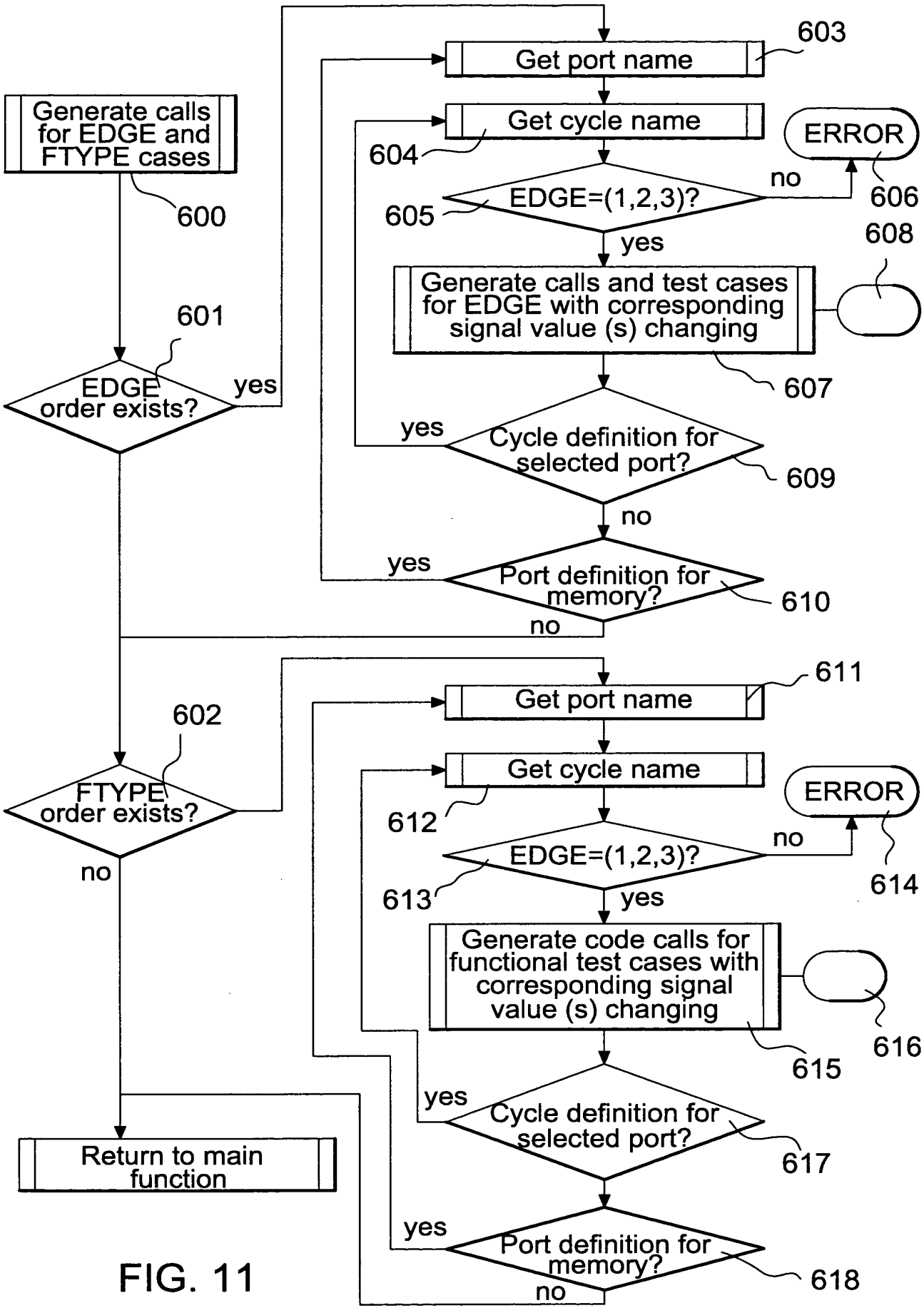

Generate calls for EDGE and FTYPE cases
600
601
EDGE order exists?
yes
602
FTYPE order exists?
no
Return to main function
603
Get port name
604
Get cycle name
605
EDGE=(1,2,3)?
no
ERROR
606
yes
608
Generate calls and test cases for EDGE with corresponding signal value (s) changing
607
yes
Cycle definition for selected port?
609
no
yes
Port definition for memory?
610
no
611
Get port name
612
Get cycle name
613
EDGE=(1,2,3)?
no
ERROR
614
yes
Generate code calls for functional test cases with corresponding signal value (s) changing
615
616
yes
Cycle definition for selected port?
617
yes
Port definition for memory?
618
no

FIG. 11

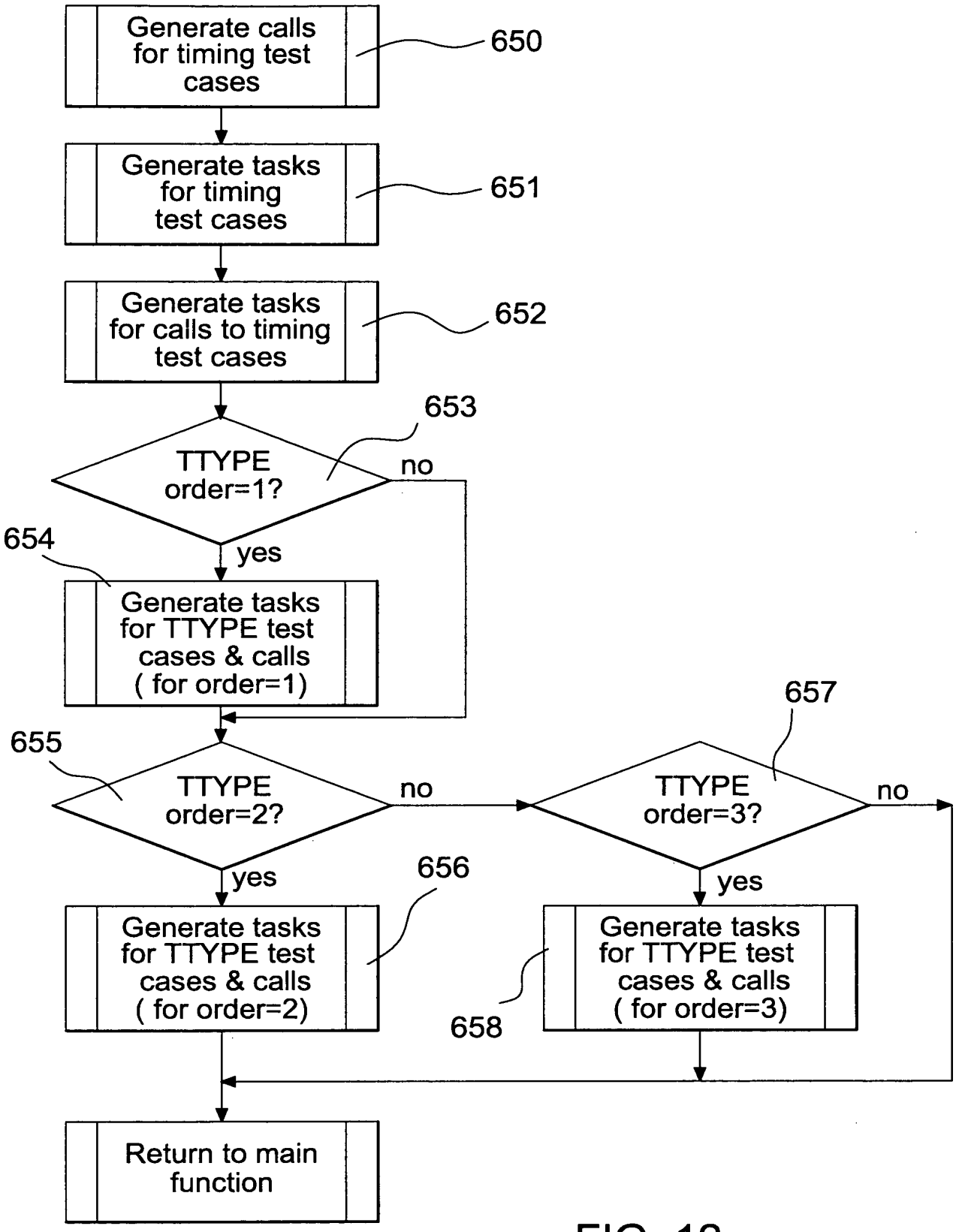

Generate calls for timing test cases
650
Generate tasks for timing test cases
651
Generate tasks for calls to timing test cases
652
TTYPE order=1?
653
no
yes
654
Generate tasks for TTYPE test cases & calls (for order=1)
655
TTYPE order=2?
no
yes
656
Generate tasks for TTYPE test cases & calls (for order=2)
657
TTYPE order=3?
no
yes
658
Generate tasks for TTYPE test cases & calls (for order=3)
Return to main function

FIG. 12

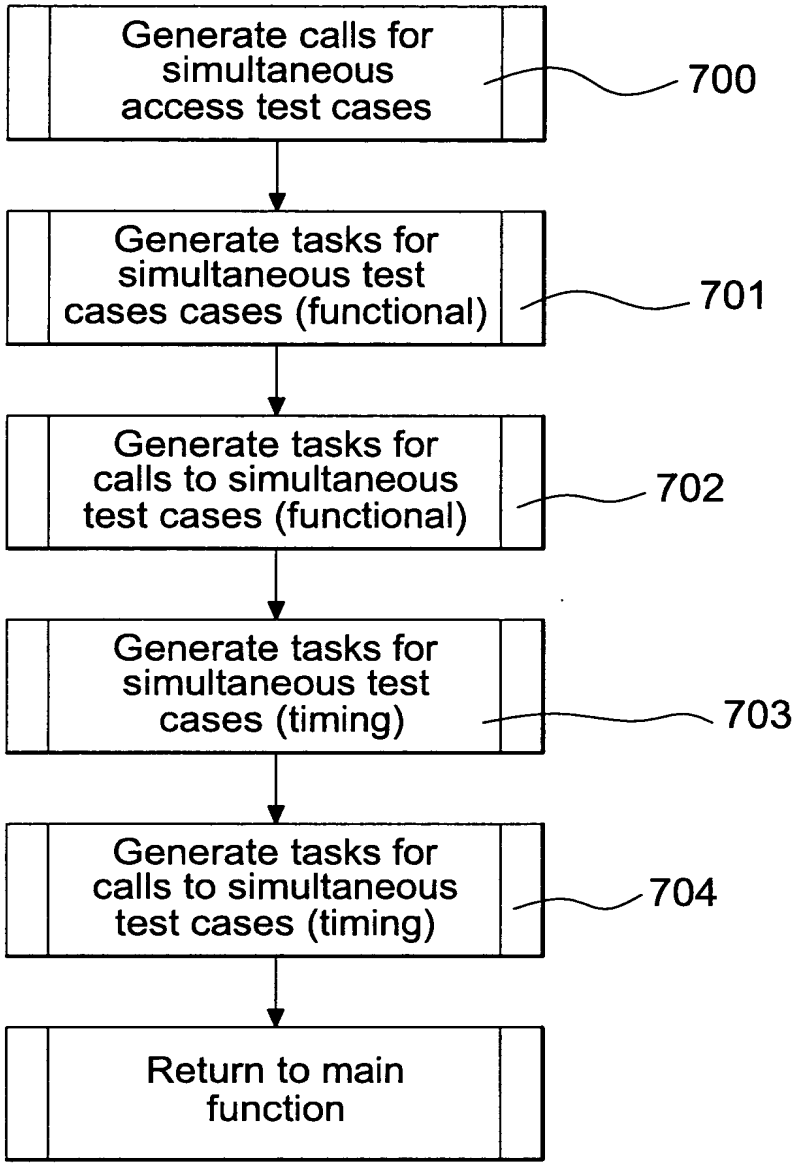

Generate calls for simultaneous access test cases
700
Generate tasks for simultaneous test cases cases (functional)
701
Generate tasks for calls to simultaneous test cases (functional)
702
Generate tasks for simultaneous test cases (timing)
703
Generate tasks for calls to simultaneous test cases (timing)
704
Return to main function

FIG. 13

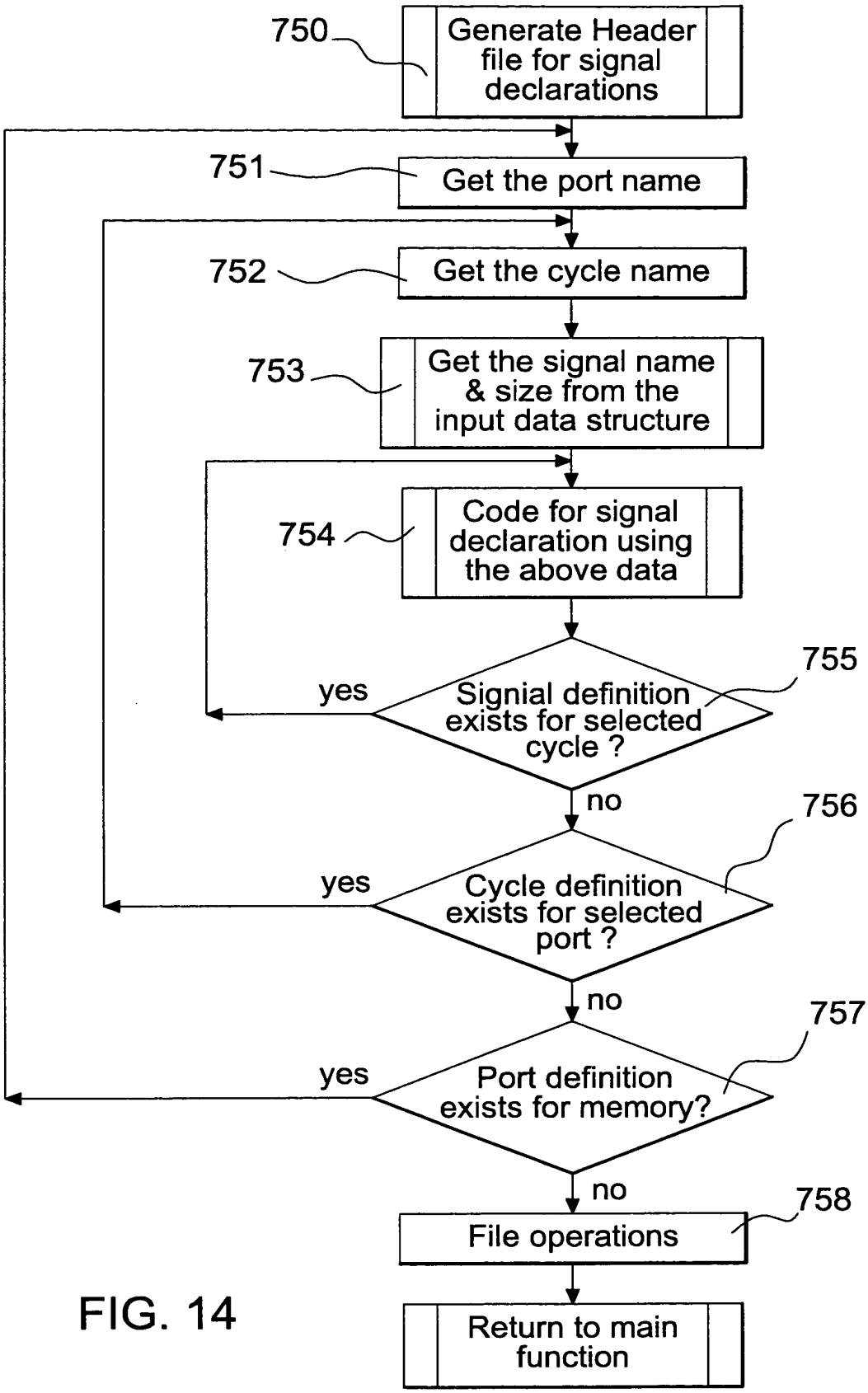

750
Generate Header file for signal declarations
751
Get the port name
752
Get the cycle name
753
Get the signal name & size from the input data structure
754
Code for signal declaration using the above data
755
Signial definition exists for selected cycle ?
yes
no
756
Cycle definition exists for selected port ?
yes
no
757
Port definition exists for memory?
yes
no
758
File operations
Return to main function

FIG. 14